

Bubblegum

Dokumentation Teil 1

Version 1.0.0, 08.07.19 | Projektteam: Joey Sciamanna, Nicola Schüepp

Inhalt

[1 Abstract 2](#_Toc479154634)

[2 Anforderungsanalyse 3](#_Toc479154635)

[2.1 User Stories 3](#_Toc479154636)

[3 Lieferumfang 4](#_Toc479154637)

[3.1 Applikation 4](#_Toc479154638)

[3.2 Dokumentation 4](#_Toc479154639)

[4 GUI Design 5](#_Toc479154640)

[4.1 MockUp 5](#_Toc479154641)

[5 OO Analyse- / Design 6](#_Toc479154642)

[5.1 Klassendiagramm 6](#_Toc479154643)

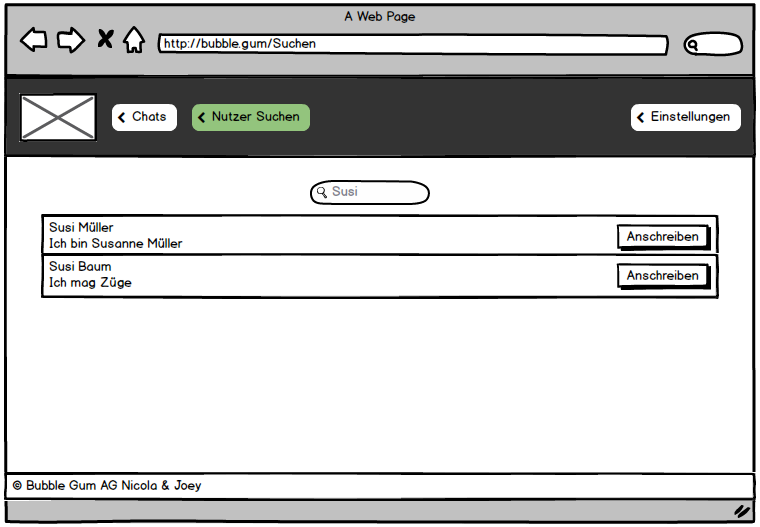
[5.2 ERM 6](#_Toc479154644)

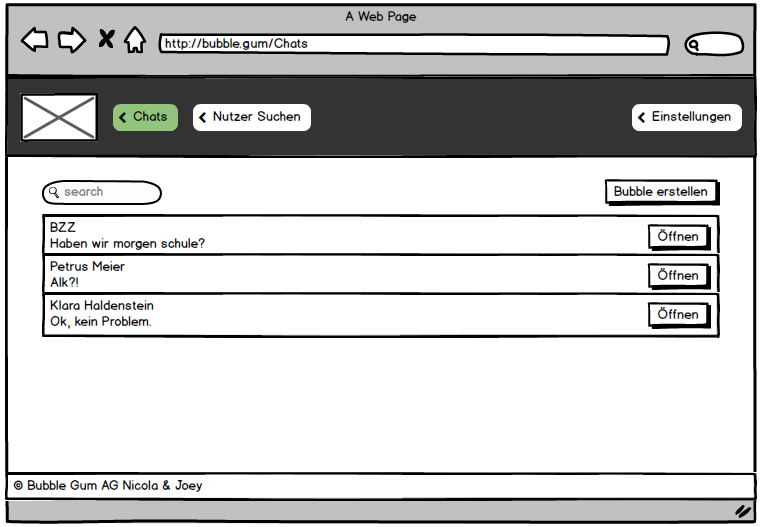
# Abstract

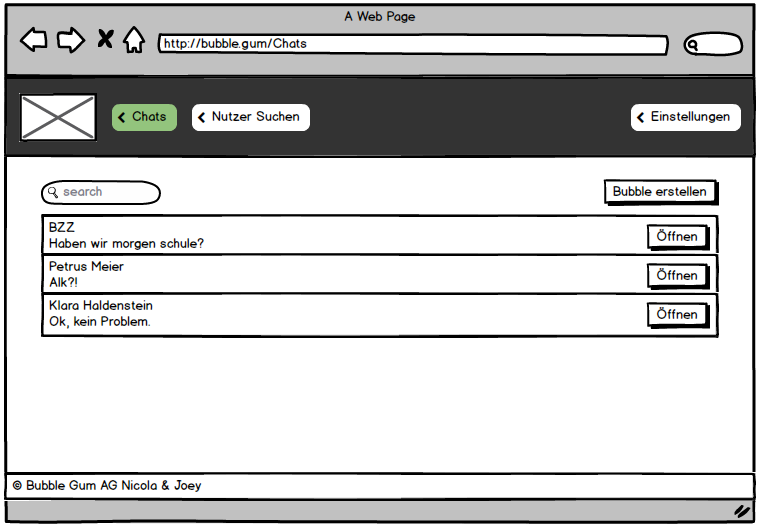
Unser Projekt umfasst einen der schönsten Online Chats im WWW. Unser Projekt nannten wir BubbleGum. BubbleGum ist ein Online Chat welcher den Nutzern ermöglicht ohne grossen Aufwand mit anderen Nutzern kurze Nachrichten auszutauschen (Quick Chat). Aber auch komplexere und Grössere Konversation in Gruppen sind mit BubbleGum in den sogenannten Bubbles möglich.

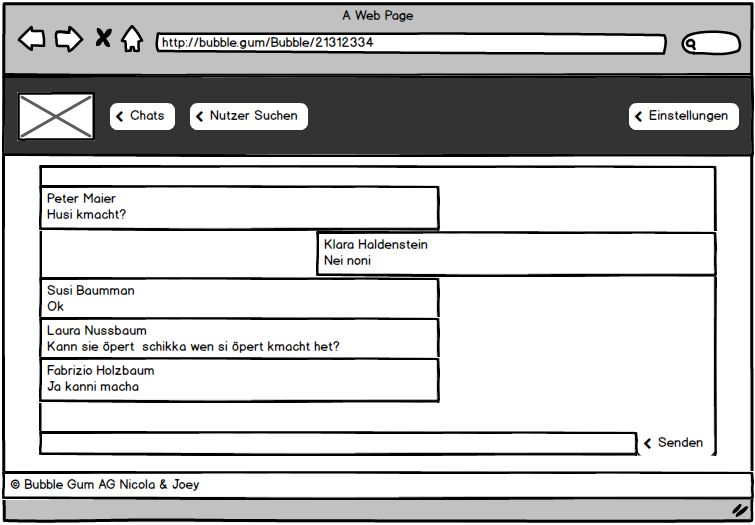
# GUI Design

## MockUp

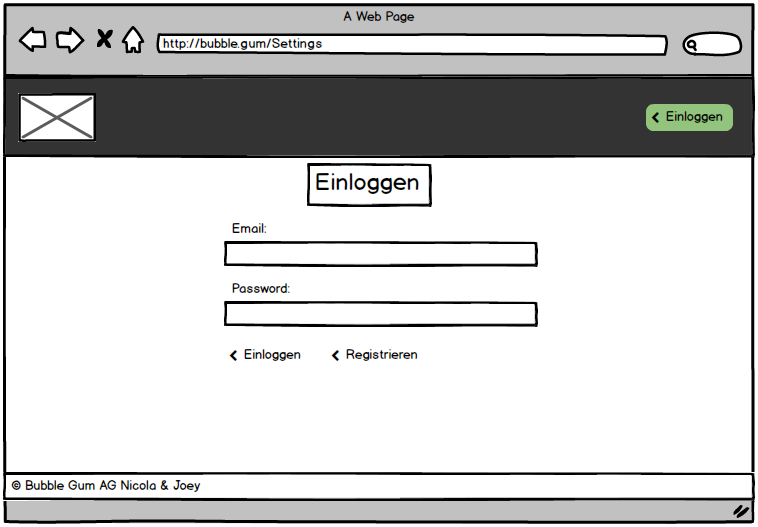


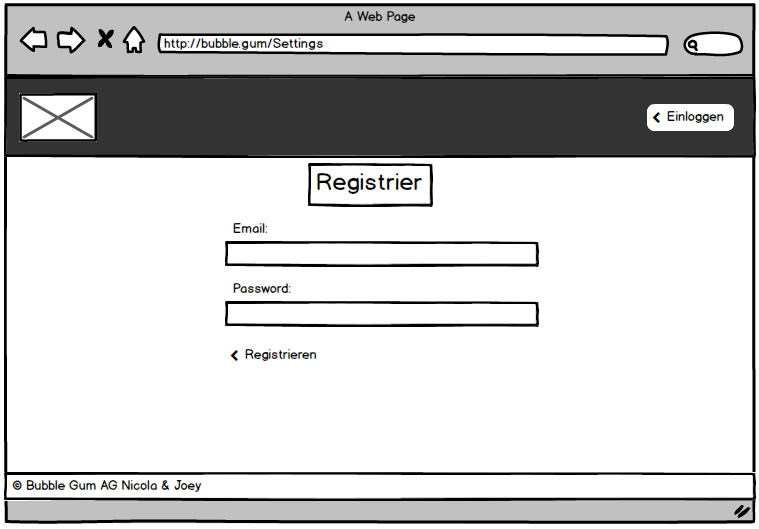






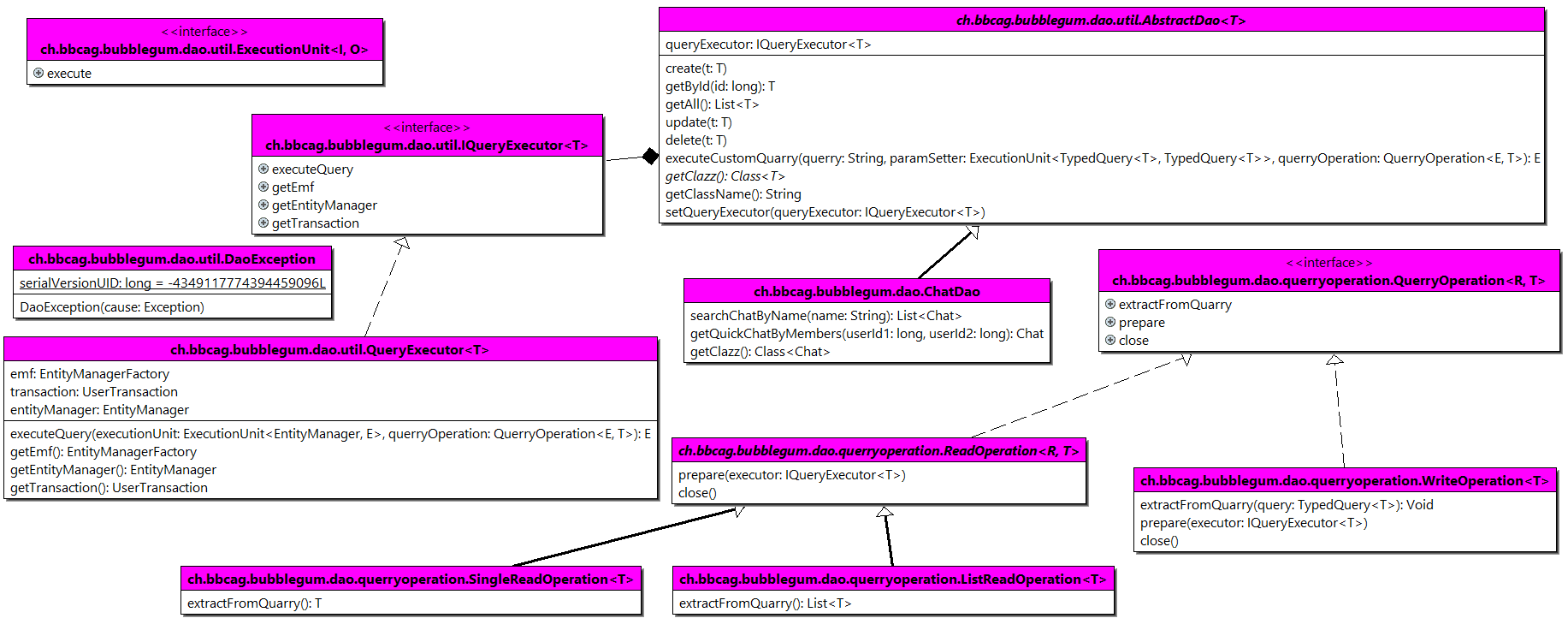






# OO Analyse- / Design

## Klassendiagramm



Klassendiagramm des DAO-Util, welches wir nutzen, um den Zugriff auf die Datenbank zu vereinfachen und Redundanz zu verhindern. Mithilfe dessen konnten wir das Ausführen von SQL Statements auf eine (lange) Zeile vermindern:  
**return** executeCustomQuarry("SELECT c FROM Chat c where c.name like :name",(q)-> q.setParameter("name", "%"+name+"%"), **new** ListReadOperation<Chat>());

Auch verschieden Resultate wie List<Chat> oder Chat konnten wir dadurch aus dem Query Extrahieren. Dies ist möglich dank Generics und den Klassen ListReadOperation (List<Chat>), SingleReadOperation (Chat), WriteOperation (Void). Ausserdem haben wir die üblichen CRUD Methoden ausgelagert, diese müssen dann nur noch im Interface des jeweiligen DAOs «freigeschaltet» werden.

## ERM

